

# Perceptive Content Database Upgrade Package for PostgreSQL

## Reference Guide

Version: Foundation 24.1

Written by: Documentation Team, R&D  
Date: July 2024

## Documentation Notice

Information in this document is subject to change without notice. The software described in this document is furnished only under a separate license agreement and may only be used or copied according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in the license agreement. This document or accompanying materials may contain certain information which is confidential information of Hyland Software, Inc. and its affiliates, and which may be subject to the confidentiality provisions agreed to by you.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. or one of its affiliates.

Hyland, HXP, OnBase, Alfresco, Nuxeo, and product names are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

© 2024 Hyland Software, Inc. and its affiliates.

The information in this document may contain technology as defined by the Export Administration Regulations (EAR) and could be subject to the Export Control Laws of the U.S. Government including for the EAR and trade and economic sanctions maintained by the Office of Foreign Assets Control as well as the export controls laws of your entity's local jurisdiction. Transfer of such technology by any means to a foreign person, whether in the United States or abroad, could require export licensing or other approval from the U.S. Government and the export authority of your entity's jurisdiction. You are responsible for ensuring that you have any required approvals prior to export.

## Table of Contents

<b>Documentation Notice</b> .....	<b>2</b>
<b>Overview</b> .....	<b>5</b>
<b>Benefits</b> .....	<b>5</b>
<b>Summary of steps</b> .....	<b>5</b>
Setup steps.....	6
Uptime steps.....	6
Downtime steps .....	6
Status procedures .....	6
<b>Storage considerations</b> .....	<b>7</b>
Database storage for the IN_INSTANCE_PROP indexes (EP1 7.4).....	7
Database storage for the IN_DOC table and indexes (EP2 7.5).....	7
Audit table indexes .....	7
<b>Create the database packages</b> .....	<b>8</b>
IN_DB_UTIL package.....	8
IN_DB_UPGRADE package.....	8
<b>Setup steps</b> .....	<b>8</b>
Create upgrade control tables .....	8
Synchronization framework (EP2 7.5).....	9
<b>Uptime steps</b> .....	<b>9</b>
<b>Synchronization steps (EP2 7.5)</b> .....	<b>9</b>
<b>Downtime steps</b> .....	<b>10</b>
Pre-downtime steps .....	10
Downtime steps .....	10
Post downtime steps .....	10
<b>View and update upgrade parameter default values</b> .....	<b>11</b>
<b>View and update synchronization parameter default values (EP2 7.5)</b> .....	<b>11</b>
<b>Logging</b> .....	<b>11</b>
<b>Troubleshooting</b> .....	<b>12</b>
<b>Cleanup after a disconnected or killed session</b> .....	<b>12</b>
<b>Upgrade and synchronization setup procedures</b> .....	<b>14</b>
Upgrade setup procedures .....	14
<i>IN_DB_UPGRADE_SP_UPGRADE_SETUP</i> .....	14
Synchronization setup procedures (EP2 7.5).....	16
<i>IN_DB_UPGRADE_SP_SYNC_SETUP</i> .....	16
<i>IN_DB_UPGRADE_SP_SYNC_SETUP_IN_DOC</i> .....	18

Synchronization procedures (EP2 7.5).....	19
<i>IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</i> .....	20
<i>IN_DB_UPGRADE_SP_SYNC_GET_STATUS</i> .....	21
<i>IN_DB_UPGRADE_SP_SYNC_RESET</i> .....	22
<i>IN_DB_UPGRADE_SP_SYNC_REMOVE</i> .....	24
Upgrade procedures (EP2 7.5) .....	24
<i>IN_DB_UPGRADE_SP_UPTIME_STEPS</i> .....	25
<i>IN_DB_UPGRADE_SP_DOWNTIME_STEPS</i> .....	27
<i>IN_DB_UPGRADE_SP_CREATE_DEST_TABLE_IN_DOC</i> .....	29
<i>IN_DB_UPGRADE_SP_LOAD_DEST_TABLE_IN_DOC</i> .....	29
<i>IN_DB_UPGRADE_SP_CREATE_INDEXES_IN_DOC</i> .....	30
Upgrade procedures.....	32
<i>IN_DB_UPGRADE_SP_UPTIME_STEPS</i> .....	32
<i>IN_DB_UPGRADE_SP_DOWNTIME_STEPS</i> .....	33
<b>Upgrade status procedures</b> .....	<b>35</b>
<i>IN_DB_UPGRADE_SP_UPTIME_GET_STATUS</i> .....	35
<i>IN_DB_UPGRADE_SP_SYNC_GET_STATUS (EP2 7.5)</i> .....	35
<b>Upgrade objects</b> .....	<b>36</b>
IN_DB_UPGRADE_CONTROL_UPGRADE table.....	36
<b>Synchronization objects (EP2 7.5)</b> .....	<b>36</b>
IN_DB_UPGRADE_CONTROL_SYNC table.....	37
SYNC_ERROR_DOC table.....	37
SYNC_HIST_DOC table.....	37
SYNC_STAGE_DOC table.....	38
SYNC_STATE_DOC table .....	38
SYNC_TRG_DOC trigger.....	39
SYNC_FUNC_DOC trigger function.....	39
SYNC_SEQ_DOC sequence .....	39
<b>Debugging</b> .....	<b>39</b>
<b>Synchronization benchmarks (EP2 7.5)</b> .....	<b>39</b>

## Overview

The following document guides you through the steps to upgrade the inuser database schema using the Perceptive Content IN\_DB\_UPGRADE package for PostgreSQL.

You can use the IN\_DB\_UPGRADE package as an alternative to the traditional database incremental scripts that were previously and still available for database schema upgrades.

This collection of procedures and functions (package) is designed to provide a means for you to upgrade your Perceptive Content database schema in a manner that facilitates the execution of the most time-consuming schema changes in uptime to reduce the duration of downtime events during database upgrades.

This package supports upgrading the inuser database schema to version 7.9.0.0 (22.2) or 7.7.0.0 (22.1 and EP4) or 7.5.0.0 (EP3 and EP2) from 7.7.0.0 or 7.5.0.0 or 7.4.0.3 or 7.2.3.0 or 7.2.0.0 or 7.1.5.2 or 7.1.4.0 or 7.1.3.3.

This upgrade package is ideal for customers with large databases who might benefit from executing the long running operations in uptime, resulting in a much shorter downtime event during the upgrade.

For upgrades with a starting schema version less than 7.5.0.0, using this package to upgrade the database schema involves the duplication of the IN\_DOC table as well as several large indexes. This requires the allocation of additional storage for the duration of the upgrade.

Certain sections of this document that include (EP2 7.5) in the heading are only applicable for upgrades where the starting schema version is less than 7.5.0.0.

## Benefits

This upgrade package provides the following benefits:

- This upgrade package provides for additional flexibility by facilitating an upgrade to 7.9.0.0 (22.2) or 7.7.0.0 (22.1 or EP4) or 7.5.0.0 (EP3 and EP2) from any previous version between 7.1.3.3 – 7.7.0.0.
- Facilitates a schema upgrade with a limited number of commands and ensures that all database incremental scripts are executed in the proper order based on the starting schema version number and the specified target schema version number.
- The 7.2.2.0 and 7.5.0.0 incremental database scripts include operations that rebuild the IN\_DOC table and its indexes and constraints. These operations are time consuming and result in an extended downtime for larger databases. This upgrade package provides a means for accomplishing these lengthy operations during uptime in the days or hours leading up to the downtime and results in a much shorter downtime event.
- This upgrade package provides you with more control over incrementally completing the uptime steps over short periods of time by leveraging the MAX\_MINUTES parameter. This parameter helps limit the scope per execution of the uptime steps by defining an end time that prevents new tasks from starting after MAX\_MINUTES has elapsed.

## Summary of steps

The following section outlines the summary of steps to upgrade your Perceptive Content database using the IN\_DB\_UPGRADE package. These steps, and the relevant stored procedures, are described in more detail later throughout this document.

**Note** The IN\_DB\_UTIL and IN\_DB\_UPGRADE packages must be created first by running the creation scripts as instructed in the [Create the database packages](#) section.

**Note** The synchronization related steps and procedures currently only apply for upgrades with a starting schema version less than 7.5.0.0. The notation of **(EP2 7.5)** within this document indicates that the section or step only applies to upgrades where the starting schema version is less than 7.5.0.0.

## Setup steps

- Use the [IN\\_DB\\_UPGRADE\\_SP\\_UPGRADE\\_SETUP](#) procedure to setup the upgrade and create the [IN\\_DB\\_UPGRADE\\_CONTROL\\_UPGRADE](#) table, which contains the default parameter values used by the various upgrade procedures.

**(EP2 7.5)** For upgrades starting on schema versions less than 7.5.0.0, this command will also call the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_SETUP](#) procedure to create the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table, which contains the default parameter values used by the various synchronization procedures. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_upgrade_setup(null, null, null, null, null, null, null, null, null);
```

## Uptime steps

Uptime steps are optional. This is ideal for larger databases where the total duration of the downtime steps can be significantly reduced by completing the uptime steps to perform the qualifying schema changes in uptime in the hours or days leading up to the downtime. For small to medium sized databases the benefit of performing the qualifying uptime steps prior to the downtime may be less beneficial.

**Note** If you choose not to execute any steps in uptime then you can skip this step and go directly to the downtime steps.

- Use the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure to execute qualifying uptime schema changes until all steps have completed. See the [status procedures](#) section for the command to display the status of qualifying uptime steps. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_steps(null, null, null, null, null);
```

- (EP2 7.5)** For upgrades starting on schema versions less than 7.5.0.0 use the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure, as needed, to synchronize and propagate rows between the IN\_DOC and NEW\_DOC tables. See the [status procedures](#) section for the command to display the status of the synchronization events. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_table_in_doc(null, null, null, null, null);
```

## Downtime steps

- Use the [IN\\_DB\\_UPGRADE\\_SP\\_DOWNTIME\\_STEPS](#) procedure to execute any remaining qualifying uptime steps and all the downtime schema changes needed to upgrade the inuser schema to the target schema version number. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_downtime_steps(null, null);
```

## Status procedures

- Use the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_GET\\_STATUS](#) procedure to check the status of all qualifying uptime steps. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_get_status(null, null);
```

**(EP2 7.5)** For upgrades starting on schema versions less than 7.5.0.0 use the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_GET\\_STATUS](#) procedure to check the synchronization status between the IN\_DOC and NEW\_DOC tables. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_get_status('in_doc', null, null);
```

## Storage considerations

### Database storage for the IN\_INSTANCE\_PROP indexes (EP1 7.4)

The following only applies to upgrades starting on schema versions prior to 7.4.0.3.

The 7.2.3.0 to 7.4.0.3 uptime steps include the creation of four new filtered indexes on the IN\_INSTANCE\_PROP table. This table is often the largest table in the Perceptive Content database so the time and storage required to create these new indexes will be considerable for larger databases. Ensure that underlying filesystems have the capacity to accommodate the expected growth.

After the four new filtered indexes have been created then the three original/old indexes will be dropped from the inuser schema. Dropping an index acquires an access exclusive lock on the table, blocking other accesses until the index drop can be completed so these steps should either be completed after hours (nights or weekends) if using [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) or during downtime.

### Database storage for the IN\_DOC table and indexes (EP2 7.5)

The following only applies to upgrades starting on schema versions prior to 7.5.0.0.

To facilitate the process of upgrading the IN\_DOC table in uptime, a copy of the IN\_DOC table is created during the uptime steps and is synchronized with the original table until all the uptime and downtime steps have been completed. This requires that sufficient free space is available to accommodate the copy of the IN\_DOC table and its indexes. Ensure that underlying filesystems have the capacity to accommodate the expected growth.

#### Additional storage considerations

- Consideration should also be given to the expected volume of new documents created during the time that the synchronization process is in place and additional storage should be allocated to account for that growth.
- During the downtime steps, the original IN\_DOC table and indexes will be replaced by the new versions created during the uptime steps. Upon completion of the upgrade downtime steps and after the original IN\_DOC table and its indexes are dropped the additional storage will be released and become available within the respective tablespaces.
- In a default implementation, the new IN\_DOC table will be in the INOW\_Data tablespace and all the indexes will be in the INOW\_Idx tablespace. You may choose to create the new version of the IN\_DOC table and its indexes in different tablespaces if desired.

### Audit table indexes

For databases with large amounts of audit data, the index changes to the IN\_AUDIT\_DETAIL (7.7.0.0) and IN\_AUDIT (7.9.0.0) tables could take a considerable amount of time during the execution of the uptime steps.

Additionally, it may be desirable to leverage the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA procedure ahead of time to delete older audit data from the audit tables before starting any of the IN\_DB\_UPGRADE steps to minimize the amount of time and storage required to facilitate these index changes.

Please reference the IN\_DB\_UTIL documentation for more information on the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA procedure.

## Create the database packages

These database packages are currently supported on PostgreSQL versions 13 and up. Please review the Perceptive Content Technical Specifications documentation for your specific version of Perceptive Content to confirm which versions of PostgreSQL are supported.

### IN\_DB\_UTIL package

The IN\_DB\_UTIL package is a collection of integrated procedures and functions that perform common database tasks that are utilized by the IN\_DB\_UPGRADE package. To create the IN\_DB\_UTIL package, complete the following step.

- Using psql, execute the following script as the inuser user.

```
PerceptiveContentDB_IN_DB_UTIL_PostgreSQL_Package_v1.0.sql
```

### IN\_DB\_UPGRADE package

The IN\_DB\_UPGRADE package is a collection of integrated procedures and functions that manage the synchronization process (EP2 7.2) and the database schema changes for both the uptime and downtime steps. This package is used to upgrade the inuser schema to version 7.9.0.0 or 7.7.0.0 or 7.5.0.0 from schema versions 7.1.3.3 through 7.7.0.0.

- Run the following script to create the IN\_DB\_UPGRADE package.
- Using psql, execute the following script as the inuser user.

```
PerceptiveContentDB_Upgrade_Package_PostgreSQL_7900.sql
```

## Setup steps

### Create upgrade control tables

The [IN\\_DB\\_UPGRADE\\_SP\\_UPGRADE\\_SETUP](#) procedure is used to create and populate the [IN\\_DB\\_UPGRADE\\_CONTROL\\_UPGRADE](#) table, which holds the default parameter values that are used if not otherwise specified when executing the upgrade procedures.

**(EP2 7.5)** For upgrades starting on schema versions less than 7.5.0.0, the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_SETUP](#) procedure is used to create the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table, which holds the default parameter values that are used if not otherwise specified when executing the synchronization procedures.

#### Upgrade and synchronization default parameter values

For all upgrades

```
TARGET_SCHEMA_VERSION '7.9.0.0'  
MAX_MINUTES           0  
FILLFACTOR            90  
TABLESPACE_NAME       'INOW_Data'
```

**(EP2 7.5)** For upgrades starting with schema versions less than 7.5.0.0 the following parameters also apply

```
BATCH_SIZE            10000  
DEFER_FK_CHECK        'NO'
```

```
KEEP_SRC_TABLE      'YES'
```

- For any parameters that you do not want to take the default value for, pass the appropriate parameter value when executing [IN\\_DB\\_UPGRADE\\_SP\\_UPGRADE\\_SETUP](#) procedure.
- To setup the upgrade, and create and populate the [IN\\_DB\\_UPGRADE\\_CONTROL\\_UPGRADE](#) table with the desired upgrade parameters, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_upgrade_setup(null, null, null, null, null, null, null, null, null);
```

## Synchronization framework (EP2 7.5)

This only applies to upgrades with a starting schema version less than 7.5.0.0.

The synchronization framework supports the propagation of data between the source (IN\_DOC) and destination (NEW\_DOC) tables, which facilitate the uptime column modifications and data population and index creation of the new version of the IN\_DOC table.

**Note** The following steps are executed automatically when [IN\\_DB\\_UPGRADE\\_SP\\_UPGRADE\\_SETUP](#) is executed if the schema version is less than 7.5.0.0 and therefore does not need to be manually executed.

- To create and populate the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_setup('in_doc', null, null, null, null, null, null, null, null);
```

- To create the synchronization framework for the IN\_DOC table, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_setup_in_doc(null, null, null, null, null, null, null, null, null);
```

## Uptime steps

The uptime steps can be executed incrementally over a period of time or all at once depending on the parameters used when executing the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure.

**Note** If you choose not to execute any steps in uptime then you can skip this procedure and go directly to the [IN\\_DB\\_UPGRADE\\_SP\\_DOWNTIME\\_STEPS](#) procedure.

- To perform qualifying uptime schema changes, execute the following procedure. This procedure accepts the MAX\_MINUTES parameter, among others. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_steps(null, null, null, null, null);
```

- To view the progress of the qualifying uptime steps, execute the following procedure. Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_get_status(null, null);
```

## Synchronization steps (EP2 7.5)

This only applies to upgrades with a starting schema version less than 7.5.0.0.

The following procedures are used to manage the synchronization portion of the upgrade. For more details, refer to the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) usage notes and parameters section. Complete the following steps as needed.

- To synchronize the data between the new version of the IN\_DOC table (NEW\_DOC) with the existing version (IN\_DOC), based on the rows in the staging table (SYNC\_STAGE\_DOC), execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_table_in_doc(null, null, null, null, null);
```

- To view the current status of the synchronization between the source (IN\_DOC) and destination (NEW\_DOC) tables, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_get_status('in_doc', null, null);
```

## Downtime steps

### Pre-downtime steps

**Important** Before executing the downtime steps, ensure the following:

- Take a full backup of the database.
- Stop all the services for the Perceptive Content Application server.
- Check for any other connections to the database and disconnect them.
  - You can use the following procedure to check for database connections:

```
SELECT * FROM in_db_util_fn_db_connections_display();
```

### Downtime steps

To execute the qualifying downtime schema changes necessary to upgrade the inuser database schema to 7.9.0.0 or 7.7.0.0 or 7.5.0.0, complete the following step.

**Note** The [IN\\_DB\\_UPGRADE\\_SP\\_DOWNTIME\\_STEPS](#) procedure calls the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure to ensure all qualifying prerequisite actions have completed if the starting schema version is less than 7.5.0.0.

- To complete the database downtime steps, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_downtime_steps(null, null);
```

- To view the progress of the uptime steps, execute the following procedure from a separate session as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_get_status(null, null);
```

### Post downtime steps

Upon successful completion of the downtime steps, proceed with the following Perceptive Content Server related upgrade steps.

1. Vacuum and analyze the database to cleanup dead rows and update database statistics.

```
vacuum (verbose, analyze);
```

2. Take a full backup of the database.
3. Complete Perceptive Content Server related upgrade steps.
4. Conduct standard user acceptance testing procedures.

## View and update upgrade parameter default values

You can query or manually update the `IN_DB_UPGRADE_CONTROL_UPGRADE` table to view or change the default values used during the upgrade related procedures.

**Note** You can also update the default upgrade parameters by re-executing the `UPGRADE_SETUP` procedure using either the default values or by specifying the desired values for the respective parameters.

- To view the current parameter defaults, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_UPGRADE;
```

- The following is an example of updating default values.

**Note** Not all columns in this table should be modified. Updating some columns can lead to unintended behavior or prevent the package from functioning properly. Ensure to only update the following parameters as needed.

The values used in the following example are not the defaults in some cases. See the parameter descriptions for each procedure in the [Synchronization and Upgrade Procedures](#) section.

```
UPDATE INUSER.IN_DB_UPGRADE_CONTROL_UPGRADE SET
  SCHEMA_TARGET = '7.9.0.0'
,DEFAULT_MAX_MINUTES = 60
,DEFAULT_FILLFACTOR = 90;
```

## View and update synchronization parameter default values (EP2 7.5)

Only applies to upgrades with a starting schema version less than 7.5.0.0.

You can query or manually update the `IN_DB_UPGRADE_CONTROL_SYNC` table to view or change the default values used during the synchronization related procedures.

**Note** You can also update the default synchronization parameters by re-executing the `IN_DB_UPGRADE_SP_SYNC_SETUP` procedure using either the default values or by specifying the desired values for the respective parameters.

- To view the current parameter defaults, execute the following query.

```
SELECT * FROM INUSER.IN_DB_UPGRADE_CONTROL_SYNC;
```

- The following is an example of updating default values.

**Note** Not all columns in this table should be modified. Updating some columns can lead to unintended behavior or prevent the package from functioning properly. Ensure to only update the following parameters as needed.

The values used in the following example are not the defaults in some cases. See the parameter descriptions for each procedure in the [Synchronization and Upgrade Procedures](#) section.

```
UPDATE INUSER.IN_DB_UPGRADE_CONTROL_SYNC SET
  DEFAULT_MAX_MINUTES = 60
,DEFAULT_BATCH_SIZE = 50000
,KEEP_SRC_TABLE = 'YES'
,DEFER_FK_CHECK = 'YES';
```

## Logging

`IN_DB_UPGRADE` package uses the `IN_DB_UTIL_SP_LOGGER` procedure to log certain actions to the `IN_DB_UTIL_LOG` table. This table will contain a log of the procedures that have been executed as well as any exceptions that occur during the execution of the `IN_DB_UTIL` and `IN_DB_UPGRADE` packages.

Please reference the IN\_DB\_UTIL documentation for more information.

To view the contents of the IN\_DB\_UTIL\_LOG table execute the query provided in the [Troubleshooting](#) section below.

## Troubleshooting

- To view active session details, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_util_sp_sessions_display();
```

- To view a report on uptime progress, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_get_status(null, null);
```

- (EP2 7.5)** To view a report on synchronization events, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_get_status('in_doc', null, null);
```

- (EP2 7.5)** To view the current state of synchronization, execute the following query. This table is queried by the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_GET\\_STATUS](#) procedure.

```
SELECT * FROM inuser.sync_state_doc;
```

- (EP2 7.5)** To view the history of synchronization activity, execute the following query. This table is queried by the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_GET\\_STATUS](#) procedure.

```
SELECT * FROM inuser.sync_hist_doc ORDER BY sync_datetime;
```

- (EP2 7.5)** To view the history of synchronization errors, execute the following query. This table is queried by the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_GET\\_STATUS](#) procedure.

```
SELECT * FROM inuser.sync_error_doc ORDER BY action_datetime;
```

- (EP2 7.5)** To view the control table for synchronization default values, execute the following query.

```
SELECT * FROM inuser.in_db_upgrade_control_sync;
```

- To view the control table for upgrade default values, execute the following query.

```
SELECT * FROM inuser.in_db_upgrade_control_upgrade;
```

- The IN\_DB\_UTIL\_LOG table contains a log of any exceptions that occur during the execution of the IN\_DB\_UTIL and IN\_DB\_UPGRADE packages.

- To view the contents of the IN\_DB\_UTIL\_LOG table execute the following query.

```
SELECT * FROM inuser.in_db_util_log ORDER BY log_id;
```

- For more in-depth troubleshooting you can execute the various procedures using the debug option by setting the **in\_debug** parameter = 'YES'.

## Cleanup after a disconnected or killed session

If a session or procedure is disconnected or killed before it is able to complete then you will likely encounter errors similar to the following the next time you try to execute one of the upgrade or synchronization procedures. If a procedure terminates in a manner that prevents the post execution cleanup procedures from running then the status for the identifier will not get reset and will prevent new operations from being executed.

**Note** If you receive errors similar to the following then the procedure is either currently being executing by another session and it should either be allowed to continue executing or if the session is known to have

been killed or disconnected then continue below to gather more information to be used in cleaning up the session.

```
WARNING: The ALL application lock resource is currently being held by another session.
NOTICE: IS_EXECUTING - There is another session already executing the in_db_upgrade_sp_uptime_steps procedure under the ALL
identifier
WARNING: The SYNC application lock resource is currently being held by another session.
NOTICE: IS_EXECUTING - There is another session already executing the in_db_upgrade_sp_uptime_steps procedure under the ALL
identifier
WARNING: The UPGRADE application lock resource is currently being held by another session.
NOTICE: IS_EXECUTING - There is another session already executing the in_db_upgrade_sp_uptime_steps procedure under the ALL
identifier
WARNING: The SYNC application lock resource for in db upgrade_sp uptime_steps is currently being held by another session.
NOTICE: IS_EXECUTING - There is another session already executing the in_db_upgrade_sp_uptime_steps procedure under the ALL
identifier
WARNING: The UPGRADE application lock resource for in_db_upgrade_sp uptime_steps is currently being held by another session.
NOTICE: IS_EXECUTING - There is another session already executing the in_db_upgrade_sp_uptime_steps procedure under the ALL
identifier
NOTICE:
NOTICE: The following session is currently holding advisory locks
NOTICE:
NOTICE: Session ID          = 36484
NOTICE: Database Name      = INOW
NOTICE: User Name           = inuser
NOTICE: Client Address      = ::1
NOTICE: Application Name    =
NOTICE: Start Time         = 2024-06-10 08:53:38.218024-05
NOTICE: State               = idle
NOTICE:
WARNING: Failed to acquire one or more advisory locks
```

You can use the following procedures to view and manually clear the IN\_DB\_UTIL framework (session attributes and callstack and Postgres advisory locks) for the SYNC and UPGRADE identifiers as needed after an unhandled exception or after manually cancelling one of the upgrade or synchronization procedures before completion.

- The following procedure can be used to see what operations are currently running for the SYNC and UPGRADE contexts. Execute the following commands as the inuser user from psql or pgAdmin 4.

```
DO $$
BEGIN
    CALL inuser.in_db_util_sp_callstack_print();
    CALL inuser.in_db_util_sp_sessions_display();
END;
$$;
```

If the operation returned by the IN\_DB\_UTIL\_SP\_SESSIONS\_DISPLAY procedure is known to have been terminated or killed then you can execute one of the following procedures to clear the status for both identifiers (ALL) or for the specified identifier (SYNC or UPGRADE).

**IMPORTANT** Do not execute any of the following procedures if any operations are still in progress as it will clear the call-stack and session attributes which will lead to undesired results and unhandled exceptions. If necessary, kill the session first then execute the following to clear the state for the identifier(s).

- To clear the session and call stack for ALL identifiers (SYNC and UPGRADE), execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_util_sp_clear_identifier('ALL', null, null, null, null);
```

- To clear session and call stack for only the SYNC identifier, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_util_sp_clear_identifier('SYNC', null, null, null, null);
```

- To clear session and call stack for only the UPGRADE identifier, execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_util_sp_clear_identifier('UPGRADE', null, null, null, null);
```

## Upgrade and synchronization setup procedures

The following procedures provide an interface for administrators to specify their preferences for the various parameters used during the synchronization and database schema upgrade steps.

The procedures validate the parameters specified during setup and retain them in the corresponding control tables for persistence. The default parameter values will be used by the procedures unless different values are specified during the execution of the procedures. The default values can be updated by running the setup procedures again or by manually updating the control tables.

**Note** Synchronization related steps and procedures currently only apply for upgrades with a starting schema version less than 7.5.0.0.

### Upgrade setup procedures

#### IN\_DB\_UPGRADE\_SP\_UPGRADE\_SETUP

The `IN_DB_UPGRADE_SP_UPGRADE_SETUP` procedure validates the default parameter values that are used during the upgrade and creates the `IN_DB_UPGRADE_CONTROL_UPGRADE` table, which is used to save the default parameter values for the duration of the upgrade. This procedure can be re-executed to update the default upgrade parameter values in the control table or to recreate the `IN_DB_UPGRADE_CONTROL_UPGRADE` table if necessary.

If the starting schema version for the upgrade is less than version 7.5.0.0 then the `IN_DB_UPGRADE_SP_UPGRADE_SETUP` procedure will automatically execute the `IN_DB_UPGRADE_SP_SYNC_SETUP` procedure to create and populate the `IN_DB_UPGRADE_CONTROL_SYNC` table to facilitate the uptime steps for the `IN_DOC` table upgrade.

#### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_upgrade_setup
(
  in_target_schema_version  IN citext_max_40,
  in_max_minutes            IN integer,
  in_fillfactor             IN integer,
  in_batch_size             IN integer,
  in_sync_defer_fk_check   IN citext_max_6,
  in_sync_keep_src_table   IN citext_max_6,
  in_tablespace            IN citext_max_64,
  in_debug                 IN citext_max_6,
  out_return_val           INOUT boolean
)
```

#### Parameters

The `IN_DB_UPGRADE_SP_UPGRADE_SETUP` procedure requires the following parameters.

Parameter	Description
<code>in_target_schema_version</code>	The target schema version for the database upgrade. This version of the upgrade package currently supports a target version of 7.5.0.0 and 7.7.0.0 and 7.9.0.0. The default value is 7.9.0.0.
<code>in_max_minutes</code>	Used to define the default maximum number of minutes that an upgrade or synchronization procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when a procedure can no longer pick up new work. This parameter helps to limit the amount of work that a

	<p>procedure is permitted to perform to help scope operations within a defined timeframe.</p> <p>This applies to the following procedures:</p> <p><a href="#">IN_DB_UPGRADE_SP_UPTIME_STEPS</a></p> <p><a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC (EP2 7.5)</a></p> <p>The default value of 0 is equal to unlimited.</p>
in_fillfactor	<p>Used to define the default fillfactor for indexes being created by the upgrade procedures. The fillfactor is a percentage that determines how full the index method will try to pack the index pages.</p> <p>The default value is 90.</p>
in_batch_size	<p><b>(EP2 7.5)</b> This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>Used to define the default batch size for the <a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</a> procedure. The batch size defines the number of rows to fetch into the synchronization work queue (cursor) for each batch.</p> <p>The default value is 10,000 rows.</p>
in_sync_defer_fk_check	<p><b>(EP2 7.5)</b> This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>Used to determine if the foreign key constraints, to and from the new IN_DOC table, should be validated after creation. If set to NO, then the SQL to manually validate the FK constraints will be displayed but not executed and must be manually executed as soon as possible after the upgrade.</p> <p>The foreign key validation portion of the downtime steps is one of the longest parts of the downtime for larger databases. If you are comfortable with the state of the synchronization process heading into the downtime, then you can reduce your overall downtime duration by setting this parameter to NO and deferring the FK validation as a manual step after the downtime has completed.</p> <p>The default value of YES means the FK constraints are validated as part of the downtime steps. This is the recommended value.</p>
in_sync_keep_src_table	<p><b>(EP2 7.5)</b> This only applies for upgrades with a starting schema version less than 7.5.0.0.</p> <p>Used to specify whether to keep the original/source IN_DOC table upon successful completion of the downtime steps. If YES, then the source table will be renamed and saved after the downtime event. If NO, then the source table will be dropped at the end of the downtime event, which will free up the storage it is consuming.</p> <p>The default value is YES to keep the original table and its PK constraint and index.</p>
in_tablespace	<p>The name of the tablespace to create the temporary upgrade and synchronization control tables within. The inuser user must have create privileges on the tablespace.</p> <p>The default value is INOW_Data.</p>
in_debug	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.</p> <p>The default value is NO.</p>
out_return_val	<p>Specifies whether the execution of the procedure was successful or not.</p>

	True = Success False = Error
--	---------------------------------

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_upgrade_setup(null, null, null, null, null, null, null, null, null);
```

```
CALL in_db_upgrade_sp_upgrade_setup('7.9.0.0', 0, 90, 10000, 'no', 'yes', 'INOW_Data', 'no', null);
```

## Synchronization setup procedures (EP2 7.5)

The following synchronization setup procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

### IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP

The IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP procedure validates the default parameter values used by the synchronization procedures and creates the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table, which is used to save the default parameter values for the duration of the upgrade. This procedure can be re-executed manually to update the default synchronization parameter values in the control table or to recreate the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table if necessary.

**Note** This procedure is automatically executed by the [IN\\_DB\\_UPGRADE\\_SP\\_UPGRADE\\_SETUP](#) procedure if the starting schema version is less than 7.5.0.0.

**Note** Manual execution of this procedure should not be necessary unless the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_REMOVE](#) procedure was manually executed to remove the synchronization framework for the IN\_DOC table.

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_setup
(
  in_table_name      IN citext_max_64,
  in_max_minutes     IN integer,
  in_batch_size      IN integer,
  in_defer_fk_check  IN citext_max_6,
  in_keep_src_table  IN citext_max_6,
  in_tablespace      IN citext_max_64,
  in_caller          IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val     INOUT boolean
)
```

### Parameters

The IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP procedure requires the following parameters.

Parameter	Description
in_table_name	The source table name for the synchronization process. Currently, only the IN_DOC table qualifies for synchronization.

in_max_minutes	<p>Used to define the default maximum number of minutes that a synchronization procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when a procedure can no longer pick up new work. This parameter helps to limit the amount of work that a procedure is permitted to perform to help scope operations within a defined timeframe.</p> <p>Applies to table synchronization during execution of the <a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</a> procedure.</p> <p>The default value of 0 is equal to unlimited and will allow the procedure to execute until all staged data has been propagated from the source table (IN_DOC) to the destination table (NEW_DOC).</p>
in_batch_size	<p>Used to define the default batch size for the <a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</a> procedure. The batch size defines the number of rows to fetch into the synchronization work queue (cursor) for each batch.</p> <p>The default value is 10,000 rows.</p>
in_defer_fk_check	<p>Used to determine if the foreign key constraints, to and from the new IN_DOC table, should be validated after creation. If set to NO, then the SQL to manually validate the FK constraints will be displayed but not executed and must be manually executed as soon as possible after the upgrade.</p> <p>The foreign key validation portion of the downtime steps is one of the longest parts of the downtime for larger databases. If you are comfortable with the state of the synchronization process heading into the downtime, then you can reduce your overall downtime duration by setting this parameter to NO and deferring the FK validation as a manual step after the downtime has completed.</p> <p>The default value of YES means the FK constraints are validated as part of the downtime steps. This is the recommended value.</p>
in_keep_src_table	<p>Used to specify whether to keep the original/source IN_DOC table upon successful completion of the downtime steps. If YES, then the source table will be renamed and saved after the downtime event. If NO, then the source table will be dropped at the end of the downtime event, which will free up the storage it is consuming.</p> <p>The default value is YES to keep the original table and its PK constraint and index.</p>
in_tablespace	<p>The name of the tablespace to create the temporary synchronization control table within. The inuser user must have create privileges on the tablespace.</p> <p>The default value is INOW_Data.</p>
in_caller	<p>The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.</p>
in_debug	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting. A value of YES when executing this procedure will also be used for any future synchronization procedures.</p> <p>The default value is NO.</p>
out_return_val	<p>Specifies whether the execution of the procedure was successful or not.</p> <p>True = Success</p> <p>False = Error</p>

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_setup('in_doc', null, null, null, null, null, null, null, null, null);
```

```
CALL in_db_upgrade_sp_sync_setup('in_doc', 0, 10000, 'no', 'yes', 'INOW_Data', null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP\_IN\_DOC

The IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP\_IN\_DOC procedure creates the synchronization framework that is specific to the IN\_DOC table. This procedure is automatically called by the IN\_DB\_UPGRADE\_SP\_SCHEMA\_UPGRADE\_7403\_7500 procedure during execution of the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure.

This procedure can be executed manually after running the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_SETUP](#) procedure to recreate the IN\_DOC synchronization framework if necessary.

**Note** Manual execution of this procedure should not be necessary unless the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_REMOVE](#) procedure was manually executed to remove the synchronization framework after being previously setup.

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_setup_in_doc
(
  in_max_minutes      IN integer,
  in_batch_size       IN integer,
  in_defer_fk_check   IN citext_max_6,
  in_keep_src_table   IN citext_max_6,
  in_tablespace       IN citext_max_64,
  in_caller           IN citext_max_64,
  in_debug            IN citext_max_6,
  out_return_val      INOUT boolean
)
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_SYNC\_SETUP\_IN\_DOC procedure requires the following parameters.

Parameter	Description
in_max_minutes	Used to define the default maximum number of minutes that the synchronization procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when a procedure can no longer pick up new work. This parameter helps to limit the amount of work that a process is permitted to perform to help scope operations within a defined timeframe.  Applies to table synchronization during execution of the <a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</a> procedure.  The default value of 0 is equal to unlimited and will allow the procedure to execute until all staged data has been propagated from the source table (IN_DOC) to the destination table (NEW_DOC).
in_batch_size	Used to define the default batch size for the <a href="#">IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC</a> procedure. The batch size defines the number of rows to fetch into the synchronization work queue (cursor) for each batch.

	The default value is 10,000 rows.
in_defer_fk_check	<p>Used to determine if the foreign key constraints, to and from the new IN_DOC table, should be validated after creation. If set to NO, then the SQL to manually validate the FK constraints will be displayed but not executed and must be manually executed as soon as possible after the upgrade.</p> <p>The foreign key validation portion of the downtime steps is one of the longest parts of the downtime for larger databases. If you are comfortable with the state of the synchronization process heading into the downtime, then you can reduce your overall downtime duration by setting this parameter to NO and deferring the FK validation as a manual step after the downtime has completed.</p> <p>The default value of YES means the FK constraints are validated as part of the downtime steps. This is the recommended value.</p>
in_keep_src_table	<p>Used to specify whether to keep the original/source IN_DOC table upon successful completion of the downtime steps. If YES, then the source table will be renamed and saved after the downtime event. If NO, then the source table will be dropped at the end of the downtime event, which will free up the storage it is consuming.</p> <p>The default value is YES to keep the original table and its PK constraint and index.</p>
in_tablespace	<p>The name of the tablespace to create the synchronization framework tables in. The inuser user must have create privileges on the tablespace.</p> <p>The default value is INOW_Data.</p>
in_caller	The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.
in_debug	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting. A value of YES when executing this procedure will also be used for any future synchronization procedures.</p> <p>The default value is NO.</p>
out_return_val	<p>Specifies whether the execution of the procedure was successful or not.</p> <p>True = Success</p> <p>False = Error</p>

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_setup_in_doc(null, null, null, null, null, null, null, null);

CALL in_db_upgrade_sp_sync_setup_in_doc(0, 10000, 'no', 'yes', 'INOW_Data', null, 'no', null);
```

## Synchronization procedures (EP2 7.5)

The following synchronization procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

## IN\_DB\_UPGRADE\_SP\_SYNC\_TABLE\_IN\_DOC

The `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure synchronizes the source table (`IN_DOC`) and the destination table (`NEW_DOC`) by propagating changed data that is recorded and staged in the `SYNC_STAGE_DOC` table.

For every row modified (`INSERT`, `UPDATE`, `DELETE`) in the source table (`IN_DOC`) the `SYNC_TRG_DOC` trigger will insert a row into the staging table to stage it for synchronization by the `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure.

During the execution of the `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure, staged rows are processed in batches according to the `in_batch_size` parameter and ordered by the `stage_id` column to promote a first in first out (FIFO) process. After a batch of rows has been successfully committed, the respective rows are deleted from the staging table. If all staged rows have been successfully processed the `SYNC_STAGE_DOC` table will be empty.

You can use the `IN_DB_UPGRADE_SP_SYNC_GET_STATUS` procedure to get a detailed report on the current state, and history, and performance of previous executions of the `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure.

The `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure accepts the `in_max_minutes` parameter to limit the amount of work that is completed per execution of the procedure. Using a smaller value for `in_max_minutes` can help to keep a synchronization event limited to a shorter timeframe per execution of the procedure. For example, you can specify 30 minutes to allow the procedure to propagate as many rows as it can within a 30-minute timeframe. After 30 minutes has elapsed, the procedure will exit gracefully.

**Note** If document creation volumes are high, the `SYNC_STAGE_DOC` staging table will also grow as rows are staged for synchronization. Be sure to execute the `IN_DB_UPGRADE_SP_SYNC_TABLE_IN_DOC` procedure periodically to synchronize the new `IN_DOC` table (`NEW_DOC`) with the current `IN_DOC` table.

**Note** You can schedule or manually execute this procedure as needed to synchronize rows between the source and destination tables at any point after synchronization setup has completed and the `NEW_DOC` table has been created, populated, and the primary key constraint has been created.

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_table_in_doc
(
  in_max_minutes      IN integer,
  in_batch_size       IN integer,
  in_caller           IN citext_max_64,
  in_debug            IN citext_max_6,
  out_return_val      INOUT boolean
)
```

### Parameters

The `SYNC_TABLE_IN_DOC` procedure requires the following parameters.

Parameter	Description
-----------	-------------

<p>in_max_minutes</p>	<p>The maximum number of minutes that the synchronization process will be allowed to begin processing a new batch of rows.</p> <p>If MAX_MINUTES is greater than 0 then a check will take place to determine if time (MAX_MINUTES) has expired before each new batch of rows is fetched for processing.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value of 0 is equal to unlimited which means synchronization will continue until all rows in the SYNC_STAGE_DOC table have been processed and the source and destination tables are fully synchronized.</p>
<p>in_batch_size</p>	<p>The batch size defines the number of rows to fetch into the synchronization work queue (cursor) for each batch.</p> <p>The default value is defined in the DEFAULT_BATCH_SIZE column of the IN_DB_UPGRADE_CONTROL_SYNC table.</p> <p>The default value is 10,000 rows.</p>
<p>in_caller</p>	<p>The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.</p>
<p>in_debug</p>	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.</p> <p>The default value is NO.</p>
<p>out_return_val</p>	<p>Specifies whether the execution of the procedure was successful or not.</p> <p>True = Success</p> <p>False = Error</p>

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_table_in_doc(null, null, null, null, null);
CALL in_db_upgrade_sp_sync_table_in_doc(0, 10000, null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS

The IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS procedure displays the status of synchronization between the source and destination tables. When executed it generates a detailed report on the current state, and history, and performance of previous executions of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure.

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_get_status
(
  in_table_name      IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val     INOUT boolean
)
```

### Parameters

The IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS procedure requires the following parameters.

Parameter	Description
in_table_name	The source table name for the synchronization process.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

## Example

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_get_status('in_doc', null, null);
```

## IN\_DB\_UPGRADE\_SP\_SYNC\_RESET

The IN\_DB\_UPGRADE\_SP\_SYNC\_RESET procedure resets the synchronization process for the specified table. This includes dropping all indexes (that exist) on the destination table and then truncating all the synchronization framework tables and the destination table then re-populating the destination table from the source table.

This procedure includes the following list of actions but can vary depending on which objects have been created as a result of previous executions of the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure.

- Disable the synchronization trigger
  - Disable SYNC\_TRG\_DOC trigger
- Reset the synchronization sequence
  - Recreate the SYNC\_SEQ\_DOC sequence
- Truncate the synchronization framework tables
  - Truncate the SYNC\_STAGE\_DOC table
  - Truncate the SYNC\_ERROR\_DOC table
  - Truncate the SYNC\_STATE\_DOC table
  - Truncate the SYNC\_HIST\_DOC table
- Drop indexes from the NEW\_DOC table, if they exist
  - Drop index NEW\_DOC\_IDX10
  - Drop index NEW\_DOC\_IDX11
  - Drop index NEW\_DOC\_IDX12
  - Drop index NEW\_DOC\_IDX2
  - Drop index NEW\_DOC\_IDX5
  - Drop index NEW\_DOC\_IDX6
  - Drop index NEW\_DOC\_IDX7

- Drop index NEW\_DOC\_IDX81
- Drop index NEW\_DOC\_IDX82
- Drop index NEW\_DOC\_IDX83
- Drop index NEW\_DOC\_IDX84
- Drop index NEW\_DOC\_IDX85
- Drop index NEW\_DOC\_IDX9
- Drop foreign key constraints to the NEW\_DOC table (should not be any)
- Drop primary key constraint from the NEW\_DOC table
  - Drop Constraint PK\_NEW\_DOC
- Truncate the NEW\_DOC table
- Re-load of the NEW\_DOC table
  - Enable the SYNC\_TRG\_DOC trigger
  - Re-populate the NEW\_DOC table from the IN\_DOC table
- Create the primary key constraint PK\_NEW\_DOC on the NEW\_DOC table

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_reset
(
  in_table_name      IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val    INOUT boolean
)
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_SYNC\_RESET procedure requires the following parameters.

Parameter	Description
in_table_name	The source table name for which to reset the synchronization framework.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_reset('in_doc', null, null);
```

## IN\_DB\_UPGRADE\_SP\_SYNC\_REMOVE

The IN\_DB\_UPGRADE\_SP\_SYNC\_REMOVE procedure removes the synchronization framework and upgrade objects which includes all of the following if they exist.

- NEW\_DOC table and indexes
- SYNC\_TRG\_DOC trigger
- SYNC\_SEQ\_DOC sequence
- SYNC\_ERROR\_DOC table
- SYNC\_STATE\_DOC table
- SYNC\_HIST\_DOC table
- SYNC\_STAGE\_DOC table

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_remove
(
  in_table_name      IN citext_max_64,
  in_caller          IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val     INOUT boolean
)
```

### Parameter

The IN\_DB\_UPGRADE\_SP\_SYNC\_REMOVE procedure requires the following parameters.

Parameter	Description
in_table_name	The source table name for which to remove the synchronization framework for.
in_caller	The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

### Example

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_remove('in_doc', null, null, null);
```

## Upgrade procedures (EP2 7.5)

There are significant schema changes that are made for upgrades with a starting schema version less than 7.5.0.0. Most of these steps can be facilitated during uptime in the days or hours leading up to the downtime steps.

Please reference the [Storage considerations](#) section for information on some of the uptime steps that will require considerable time and storage.

## IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS

The IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS procedure will execute all the necessary procedures that contain qualifying uptime steps, in the order they should be executed. To minimize the impact of executing the uptime steps, it is recommended to execute this procedure during periods of low activity such as nights or weekends.

This procedure accepts the **in\_max\_minutes** parameter to impose a limit on the amount of work that is completed per execution of the procedure. The use of this parameter can help to facilitate the incremental creation of the qualifying objects during uptime in the days or hours leading up to the downtime.

**Note** You can schedule this procedure or manually execute it to incrementally execute the uptime steps at any point after upgrade setup and prior to the execution of the downtime steps.

**Note** For databases with significant amounts of audit data, the index changes to the IN\_AUDIT and IN\_AUDIT\_DETAIL tables could take a while. If necessary, consider using the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA procedure to cleanup old audit data prior to starting any uptime or downtime steps.

This procedure executes the following functions and procedures in the following order:

- in\_db\_util\_fn\_get\_db\_info
- in\_db\_util\_fn\_db\_connections\_display
- in\_db\_util\_fn\_db\_connections\_count
- in\_db\_upgrade\_sp\_uptime\_steps
  - in\_db\_upgrade\_sp\_schema\_upgrade\_7220\_7230
    - Create the IN\_PHSOB.PHSOB\_IDX10 index
  - in\_db\_upgrade\_sp\_schema\_upgrade\_7230\_7403
    - Create new filtered indexes on the IN\_INSTANCE\_PROP table
      - INSTANCE\_PROP\_IDX\_01
      - INSTANCE\_PROP\_IDX\_02
      - INSTANCE\_PROP\_IDX\_03
      - INSTANCE\_PROP\_IDX\_04
    - After all the new filtered indexes have been created then the original/old indexes will be dropped from the schema.
      - Note** Dropping an index acquires an access exclusive lock on the table, blocking other accesses until the index drop can be completed.
    - INST\_PROP\_IDX\_02 (Previously instance\_prop\_idx\_02)
    - INST\_PROP\_IDX\_03
    - INST\_PROP\_IDX\_04
  - in\_db\_upgrade\_sp\_schema\_upgrade\_7403\_7500
    - IN\_DOC table rebuild and synchronization
      - in\_db\_upgrade\_sp\_sync\_setup\_in\_doc
      - in\_db\_upgrade\_sp\_create\_dest\_table\_in\_doc

- in\_db\_upgrade\_sp\_load\_dest\_table\_in\_doc
  - in\_db\_upgrade\_sp\_create\_indexes\_in\_doc
- in\_db\_upgrade\_sp\_schema\_upgrade\_7500\_7700
  - Recreate the IN\_AUDIT\_DETAIL.AUDIT\_DETAIL\_IDX03 index
- in\_db\_upgrade\_sp\_schema\_upgrade\_7700\_7900
  - Create the IN\_ACTION\_GROUP.ACTION\_GROUP\_IDX\_01 index
  - Create the IN\_AUDIT.AUDIT\_IDX\_01 index
  - Create the IN\_SC\_CONNECTION.SC\_CONNECTION\_IDX03 index
  - Create the IN\_SC\_SESSION.SC\_SESSION\_IDX04 index
  - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_07 index
  - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_08 index
- in\_db\_upgrade\_sp\_sync\_table\_in\_doc
  - Propagate staged rows from IN\_DOC into NEW\_DOC
- in\_db\_upgrade\_sp\_sync\_get\_status

## Procedure definition

```

PROCEDURE inuser.in_db_upgrade_sp_uptime_steps
(
  in_max_minutes      IN integer,
  in_fillfactor       IN integer,
  in_caller           IN citext_max_64,
  in_debug            IN citext_max_6,
  out_return_val      INOUT boolean
)
  
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS procedure requires the following parameters.

Parameter	Description
in_max_minutes	<p>The maximum number of minutes that the uptime procedure will be allowed to begin new operations.</p> <p>If in_max_minutes is greater than 0 then a check will take place to determine if time (MAX_MINUTES) has expired before each new operation is started.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the <a href="#">IN_DB_UPGRADE_CONTROL_UPGRADE</a> table.</p> <p>The default value of 0 is equal to unlimited which means the procedure will continue until all qualifying uptime steps have completed.</p>
in_fillfactor	<p>Used to define the default fillfactor for indexes being created by the upgrade procedures. The fillfactor is a percentage that determines how full the index method will try to pack the index pages.</p> <p>The default value is 90.</p>

in_caller	The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_steps(null, null, null, null, null);
CALL in_db_upgrade_sp_uptime_steps(0, 90, null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS

The IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS procedure calls all the other uptime and downtime related procedures in the order they should be executed to upgrade the inuser database schema to the target schema version.

The downtime procedure will not execute if there are more than five other connections to the database or if there are any active database connections. Ensure that all other database connections have been disconnected and there are no other active sessions.

For databases with large amounts of audit data, the index changes to the IN\_AUDIT and IN\_AUDIT\_DETAIL tables could take a while so consider leveraging the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure to do this work in uptime in the days leading up to the downtime. Additionally, it may be desirable to leverage the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA stored procedure ahead of time to delete older audit data from the tables before starting any of the IN\_DB\_UPGRADE steps.

When IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS is executed, the **in\_max\_minutes** parameter is hard coded to 0 which is equal to unlimited to ensure that all steps run to completion.

This procedure executes the following functions and procedures in the following order:

- in\_db\_util\_fn\_get\_db\_info
- in\_db\_util\_fn\_db\_connections\_display
- in\_db\_util\_fn\_db\_connections\_count
- in\_db\_upgrade\_sp\_uptime\_steps (see the [uptime\\_steps](#) for more details)
- in\_db\_upgrade\_sp\_schema\_upgrade\_7133\_7140
- in\_db\_upgrade\_sp\_schema\_upgrade\_7140\_7152
- in\_db\_upgrade\_sp\_schema\_upgrade\_7152\_7201
- in\_db\_upgrade\_sp\_schema\_upgrade\_7220\_7230
- in\_db\_upgrade\_sp\_schema\_upgrade\_7230\_7403
- in\_db\_upgrade\_sp\_schema\_upgrade\_7403\_7500

- IN\_DOC table rebuild and synchronization
  - in\_db\_upgrade\_sp\_create\_fk\_in\_doc
  - in\_db\_util\_sp\_get\_index\_ddl
  - in\_db\_util\_sp\_drop\_index
  - in\_db\_upgrade\_sp\_sync\_remove
- in\_db\_upgrade\_sp\_schema\_upgrade\_7500\_7700
- in\_db\_upgrade\_sp\_schema\_upgrade\_7700\_7900
- in\_db\_util\_fn\_schema\_get\_hist

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_downtime_steps
(
  in_debug          IN citext_max_6,
  out_return_val    INOUT boolean
)
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS procedure requires the following parameters.

Parameter	Description
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL inuser.in_db_upgrade_sp_downtime_steps('null', null);
```

```
CALL inuser.in_db_upgrade_sp_downtime_steps('no', null);
```

**Note** If `in_sync_defer_fk_check = 'YES'` was specified during the upgrade setup or the [IN\\_DB\\_UPGRADE\\_CONTROL\\_SYNC](#) table was manually updated to set `DEFER_FK_CHECK = 'YES'`, then you will need to manually execute the following commands to validate the foreign key constraints to and from the IN\_DOC table.

```
ALTER TABLE inuser.in_doc VALIDATE CONSTRAINT fk_doc_drawer_id;
ALTER TABLE inuser.in_doc VALIDATE CONSTRAINT fk_doc_instance_id;
ALTER TABLE inuser.in_doc VALIDATE CONSTRAINT fk_d_doc_type_id;
ALTER TABLE inuser.in_doc_kw VALIDATE CONSTRAINT fk_dk_doc_id;
ALTER TABLE inuser.in_rc_record_doc VALIDATE CONSTRAINT fk_rcrd_d_id;
ALTER TABLE inuser.in_sig VALIDATE CONSTRAINT fk_s_doc_id;
ALTER TABLE inuser.in_task VALIDATE CONSTRAINT fk_t_doc_id;
ALTER TABLE inuser.in_version VALIDATE CONSTRAINT fk_v_doc_id;
```

The following procedure can be used to check the status of the foreign key constraints to and from the IN\_DOC table. They should all indicate true for the Validated column.

```
SELECT inuser.in_db_util_fn_get_fk_info('in_doc', null, null, null);
```

## IN\_DB\_UPGRADE\_SP\_CREATE\_DEST\_TABLE\_IN\_DOC

The IN\_DB\_UPGRADE\_SP\_CREATE\_DEST\_TABLE\_IN\_DOC procedure creates the destination table used by the synchronization process for the IN\_DOC table. This version of the IN\_DOC table represents its structure as of schema version 7.5.0.0 and up. This procedure is executed indirectly by the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure or can be executed manually after running the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_SETUP\\_IN\\_DOC](#) procedure.

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_create_dest_table_in_doc
(
  in_tablespace      IN citext_max_64,
  in_caller          IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val     INOUT boolean
)
```

### Parameters

The IN\_DB\_UPGRADE\_SP\_CREATE\_DEST\_TABLE\_IN\_DOC procedure requires the following parameters.

Parameter	Description
in_tablespace	The name of the tablespace to create the new IN_DOC table in. The inuser user must have create privileges on the tablespace. The default value is INOW_Data.
in_caller	The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting. The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not. True = Success False = Error

### Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_create_dest_table_in_doc (null, null, null, null);
```

```
CALL in_db_upgrade_sp_create_dest_table_in_doc ('INOW_Data', null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_LOAD\_DEST\_TABLE\_IN\_DOC

The IN\_DB\_UPGRADE\_SP\_LOAD\_DEST\_TABLE\_IN\_DOC procedure copies all rows from the source table (IN\_DOC) to the destination table (NEW\_DOC). This procedure is indirectly executed by the

[IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure or can be executed manually after running [IN\\_DB\\_UPGRADE\\_SP\\_CREATE\\_DEST\\_TABLE\\_IN\\_DOC](#).

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_load_dest_table_in_doc
(
  in_caller          IN citext_max_64,
  in_debug           IN citext_max_6,
  out_return_val     INOUT boolean
)
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_LOAD\_DEST\_TABLE\_IN\_DOC procedure requires the following parameters.

Parameter	Description
in_caller	The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_load_dest_table_in_doc(null, null, null);
CALL in_db_upgrade_sp_load_dest_table_in_doc(null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_CREATE\_INDEXES\_IN\_DOC

The IN\_DB\_UPGRADE\_SP\_CREATE\_INDEXES\_IN\_DOC procedure creates all the indexes on the new IN\_DOC table (NEW\_DOC). This procedure is called by the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure or can be executed manually after [IN\\_DB\\_UPGRADE\\_SP\\_CREATE\\_DEST\\_TABLE\\_IN\\_DOC](#), and [IN\\_DB\\_UPGRADE\\_SP\\_LOAD\\_DEST\\_TABLE\\_IN\\_DOC](#) has been executed.

This procedure accepts the **in\_max\_minutes** parameter to limit the amount of work that is completed per execution of the procedure and can help to facilitate the incremental creation of the required indexes during uptime in the days or hours leading up to the downtime with minimal impact to the rest of the system.

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_create_indexes_in_doc
(
  in_max_minutes     IN integer,
  in_fillfactor      IN integer,
  in_tablespace      IN citext_max_64,
```

```

in_caller          IN citext_max_64,
in_debug          IN citext_max_6,
out_return_val    INOUT boolean
)
    
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_CREATE\_INDEXES\_IN\_DOC procedure requires the following parameters.

Parameter	Description
in_max_minutes	<p>The maximum number of minutes that the procedure is permitted to start new operations. When greater than 0, the parameter is used to set an end time that defines when the procedure can no longer pick up new work. This parameter helps to limit the amount of work that the procedure is permitted to perform to help scope operations within a limited timeframe.</p> <p>If in_max_minutes is greater than 0 then a check will take place to determine if time has expired before creating the next index on the list.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the <a href="#">IN_DB_UPGRADE_CONTROL_UPGRADE</a> table.</p> <p>The default value of 0 is equal to unlimited which means the procedure will continue running until all the indexes have been created.</p>
in_fillfactor	<p>The fillfactor percentage to use when creating the indexes. The fillfactor is a percentage that determines how full the index method will try to pack the index pages.</p> <p>The default value is 90.</p>
in_tablespace	<p>The name of an existing tablespace to create the indexes in. The inuser user must have create privileges on the specified tablespace.</p> <p>The default value is INOW_Idx.</p>
in_caller	<p>The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.</p>
in_debug	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.</p> <p>The default value is NO.</p>
out_return_val	<p>Specifies whether the execution of the procedure was successful or not.</p> <p>True = Success False = Error</p>

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```

CALL in_db_upgrade_sp_create_indexes_in_doc(null, null, null, null, null, null);
CALL in_db_upgrade_sp_create_indexes_in_doc(60, 90, 'INOW_Idx', null, 'no', null);
    
```

## Upgrade procedures

The following applies to upgrades with a starting schema version of 7.5.0.0 and up. For upgrades starting with a schema version less than 7.5.0.0, refer to the [Upgrade Procedures \(EP2 7.5\)](#) section.

### IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS

The IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS procedure will execute all the necessary procedures that contain qualifying uptime steps, in the order they should be executed. To minimize the impact of executing the uptime steps, it is recommended to execute this procedure during periods of low activity such as nights or weekends.

This procedure accepts the **in\_max\_minutes** parameter to impose a limit on the amount of work that is completed per execution of the procedure. The use of this parameter can help to facilitate the incremental creation of the qualifying objects during uptime in the days or hours leading up to the downtime.

**Note** You can schedule this procedure or manually execute it as necessary to incrementally execute the uptime steps at any point after upgrade setup and prior to the execution of the downtime steps.

**Note** For databases with significant amounts of audit data, the index changes to the IN\_AUDIT and IN\_AUDIT\_DETAIL tables could take a while. If necessary, consider using the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA procedure to cleanup old audit data prior to starting any uptime or downtime steps.

This procedure executes the following procedures in this order:

- in\_db\_upgrade\_sp\_schema\_upgrade\_7500\_7700
  - Recreate the IN\_AUDIT\_DETAIL.AUDIT\_DETAIL\_IDX03 index
- in\_db\_upgrade\_sp\_schema\_upgrade\_7700\_7900
  - Create the IN\_ACTION\_GROUP.ACTION\_GROUP\_IDX\_01 index
  - Create the IN\_AUDIT.AUDIT\_IDX\_01 index
  - Create the IN\_SC\_CONNECTION.SC\_CONNECTION\_IDX03 index
  - Create the IN\_SC\_SESSION.SC\_SESSION\_IDX04 index
  - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_07 index
  - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_08 index

### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_uptime_steps
(
  in_max_minutes      IN integer,
  in_fillfactor       IN integer,
  in_caller           IN citext_max_64,
  in_debug            IN citext_max_6,
  out_return_val      INOUT boolean
)
```

### Parameters

The IN\_DB\_UPGRADE\_SP\_UPTIME\_STEPS procedure requires the following parameters.

Parameter	Description
-----------	-------------

in_max_minutes	<p>The maximum number of minutes that the uptime procedure will be allowed to begin new operations.</p> <p>If in_max_minutes is greater than 0 then a check will take place to determine if time (MAX_MINUTES) has expired before each new operation is started.</p> <p>The default value is defined in the DEFAULT_MAX_MINUTES column of the <a href="#">IN_DB_UPGRADE_CONTROL_UPGRADE</a> table.</p> <p>The default value of 0 is equal to unlimited which means the procedure will continue until all qualifying uptime steps have completed.</p>
in_fillfactor	<p>Used to define the default fillfactor for indexes being created by the upgrade procedures. The fillfactor is a percentage that determines how full the index method will try to pack the index pages.</p> <p>The default value is 90.</p>
in_caller	<p>The name of the procedure calling this procedure. This is only used when the procedure is called by another procedure and not necessary when manually executed.</p>
in_debug	<p>Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.</p> <p>The default value is NO.</p>
out_return_val	<p>Specifies whether the execution of the procedure was successful or not.</p> <p>True = Success</p> <p>False = Error</p>

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_steps(null, null, null, null, null);
CALL in_db_upgrade_sp_uptime_steps(0, 90, null, 'no', null);
```

## IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS

The IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS procedure calls all the other uptime and downtime related procedures in the order they should be executed to upgrade the inuser database schema to the target schema version.

The downtime procedure will not execute if there are more than five other connections to the database or if there are any active database connections. Ensure that all other database connections have been disconnected and there are no other active sessions.

For databases with large amounts of audit data, the index changes to the IN\_AUDIT and IN\_AUDIT\_DETAIL tables could take a while so consider leveraging the [IN\\_DB\\_UPGRADE\\_SP\\_UPTIME\\_STEPS](#) procedure to do this work in uptime in the days leading up to the downtime. Additionally, it may be desirable to leverage the IN\_DB\_UTIL\_SP\_CLEANUP\_AUDIT\_DATA stored procedure ahead of time to delete older audit data from the tables before starting any of the IN\_DB\_UPGRADE steps.

When IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS is executed, the **in\_max\_minutes** parameter is hard coded to 0 which is equal to unlimited to ensure that all steps run to completion.

This procedure executes the following functions and procedures in the following order:

- in\_db\_util\_fn\_get\_db\_info
- in\_db\_util\_fn\_db\_connections\_display
- in\_db\_util\_fn\_db\_connections\_count
- in\_db\_upgrade\_sp\_uptime\_steps
  - in\_db\_upgrade\_sp\_schema\_upgrade\_7500\_7700
    - Recreate the IN\_AUDIT\_DETAIL.AUDIT\_DETAIL\_IDX03 index
  - in\_db\_upgrade\_sp\_schema\_upgrade\_7700\_7900
    - Create the IN\_ACTION\_GROUP.ACTION\_GROUP\_IDX\_01 index
    - Create the IN\_AUDIT.AUDIT\_IDX\_01 index
    - Create the IN\_SC\_CONNECTION.SC\_CONNECTION\_IDX03 index
    - Create the IN\_SC\_SESSION.SC\_SESSION\_IDX04 index
    - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_07 index
    - Create the IN\_WF\_QUEUE.WF\_QUEUE\_IDX\_08 index
- in\_db\_upgrade\_sp\_schema\_upgrade\_7500\_7700
- in\_db\_upgrade\_sp\_schema\_upgrade\_7700\_7900
- in\_db\_util\_fn\_schema\_get\_hist

## Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_downtime_steps
(
  in_debug          IN citext_max_6,
  out_return_val    INOUT boolean
)
```

## Parameters

The IN\_DB\_UPGRADE\_SP\_DOWNTIME\_STEPS procedure requires the following parameters.

Parameter	Description
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting. The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not. True = Success False = Error

## Examples

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL inuser.in_db_upgrade_sp_downtime_steps('null', null);
CALL inuser.in_db_upgrade_sp_downtime_steps('no', null);
```

## Upgrade status procedures

### IN\_DB\_UPGRADE\_SP\_UPTIME\_GET\_STATUS

The IN\_DB\_UPGRADE\_SP\_UPTIME\_GET\_STATUS procedure displays the status of the qualifying uptime steps. It will list the database indexes that qualify for the uptime steps and the completion status for each index.

#### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_uptime_get_status
(
  in_debug          IN citext_max_6,
  out_return_val    INOUT boolean
)
```

#### Parameters

The IN\_DB\_UPGRADE\_SP\_UPTIME\_GET\_STATUS procedure requires the following parameters.

Parameter	Description
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting. The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not. True = Success False = Error

#### Example

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_uptime_get_status(null, null);
```

### IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS (EP2 7.5)

This procedure is only used for upgrades with a starting schema version less than 7.5.0.0.

The IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS procedure displays the status of synchronization between the source and destination tables. When executed it generates a detailed report on the current state, and history, and performance of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure.

#### Procedure definition

```
PROCEDURE inuser.in_db_upgrade_sp_sync_get_status
(
  in_table_name     IN citext_max_64,
  in_debug          IN citext_max_6,
  out_return_val    INOUT boolean
)
```

#### Parameters

The IN\_DB\_UPGRADE\_SP\_SYNC\_GET\_STATUS procedure requires the following parameters.

Parameter	Description
in_table_name	The source table name for the synchronization process.
in_debug	Used to specify if additional debugging information should be displayed during execution, to help with troubleshooting.  The default value is NO.
out_return_val	Specifies whether the execution of the procedure was successful or not.  True = Success False = Error

### Example

Execute the following procedure as the inuser user from psql or pgAdmin 4.

```
CALL in_db_upgrade_sp_sync_get_status('in_doc', null, null);
```

## Upgrade objects

### IN\_DB\_UPGRADE\_CONTROL\_UPGRADE table

This table contains the default parameter values that an upgrade procedure will use if not otherwise specified when a procedure is executed.

This table is created and populated during execution of IN\_DB\_UPGRADE\_SP\_UPGRADE\_SETUP and can be manually updated as needed to adjust certain default run-time parameters.

```
+-----+-----+-----+-----+-----+
| Column | Type | Collation | Nullable | Default |
+-----+-----+-----+-----+-----+
| database_name | citext_max_128 | | not null | |
| schema_name | citext_max_128 | | not null | |
| schema_target | citext_max_40 | | not null | |
| schema_version | citext_max_40 | | not null | |
| schema_base | citext_max_40 | | not null | |
| identifier_upgrade | citext_max_128 | | not null | |
| identifier_sync | citext_max_128 | | not null | |
| default_max_minutes | integer | | not null | |
| default_max_dop | integer | | not null | |
| default_fillfactor | integer | | not null | |
| default_batch_size | integer | | not null | |
| is_sync_upgrade | citext_max_6 | | not null | |
| debug | citext_max_6 | | not null | |
+-----+-----+-----+-----+-----+
Indexes:
 "in_db_upgrade_control_upgrade_pk" PRIMARY KEY, btree (database_name, schema_name), tablespace "INOW_Data"
Check constraints:
 "in_db_upgrade_control_upgrade_check_batch_size" CHECK (default_batch_size >= 0)
 "in_db_upgrade_control_upgrade_check_debug" CHECK (debug::citext = ANY (ARRAY['Y'::citext, 'YES'::citext, 'N'::citext, 'NO'::citext]))
 "in_db_upgrade_control_upgrade_check_fillfactor" CHECK (default_fillfactor >= 50 AND default_fillfactor <= 100)
 "in_db_upgrade_control_upgrade_check_max_dop" CHECK (default_max_dop >= 0)
 "in_db_upgrade_control_upgrade_check_max_minutes" CHECK (default_max_minutes >= 0)
 "in_db_upgrade_control_upgrade_check_sync_identifier" CHECK (identifier_sync::citext = 'sync'::citext)
 "in_db_upgrade_control_upgrade_check_sync_upgrade" CHECK (is_sync_upgrade::citext = ANY (ARRAY['YES'::citext, 'NO'::citext]))
 "in_db_upgrade_control_upgrade_check_upgrade_identifier" CHECK (identifier_upgrade::citext = 'upgrade'::citext)
Tablespace: "INOW_Data"
```

## Synchronization objects (EP2 7.5)

The following procedures are currently only used for upgrades with a starting schema version less than 7.5.0.0.

## IN\_DB\_UPGRADE\_CONTROL\_SYNC table

This table contains the default parameter values that a synchronization procedure will use if not otherwise specified when a procedure is executed.

This table is created and populated during execution of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_SETUP](#) procedure and can be manually updated as needed to adjust certain run-time parameters.

Column	Type	Collation	Nullable	Default
schema_name	citext_max_128		not null	
source_table_name	citext_max_128		not null	
new_table_name	citext_max_128		not null	
save_table_name	citext_max_128		not null	
stage_table_name	citext_max_128		not null	
error_table_name	citext_max_128		not null	
state_table_name	citext_max_128		not null	
hist_table_name	citext_max_128		not null	
sequence_name	citext_max_128		not null	
function_name	citext_max_128		not null	
trigger_name	citext_max_128		not null	
keep_src_table	citext_max_6		not null	
defer_fk_check	citext_max_6		not null	
default_max_minutes	integer		not null	
default_max_dop	integer		not null	
default_batch_size	integer		not null	
debug	citext_max_6		not null	

Indexes:  
 "pk\_in\_db\_upgrade\_control\_sync" PRIMARY KEY, btree (schema\_name, source\_table\_name), tablespace "INOW\_Data"

Check constraints:  
 "in\_db\_upgrade\_control\_sync\_check\_batch\_size" CHECK (default\_batch\_size > 0)  
 "in\_db\_upgrade\_control\_sync\_check\_debug" CHECK (debug::citext = ANY (ARRAY['Y':citext, 'YES':citext, '':citext, 'NO':citext]))  
 "in\_db\_upgrade\_control\_sync\_check\_max\_defer\_fk\_check" CHECK (defer\_fk\_check::citext = ANY (ARRAY['YES':citext, 'NO':citext]))  
 "in\_db\_upgrade\_control\_sync\_check\_max\_dop" CHECK (default\_max\_dop >= 0)  
 "in\_db\_upgrade\_control\_sync\_check\_max\_keep\_src\_table" CHECK (keep\_src\_table::citext = ANY (ARRAY['YES':citext, 'NO':citext]))  
 "in\_db\_upgrade\_control\_sync\_check\_max\_minutes" CHECK (default\_max\_minutes >= 0)

Tablespace: "INOW\_Data"

## SYNC\_ERROR\_DOC table

This table contains any errors that are encountered during execution of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure.

Column	Type	Collation	Nullable	Default
stage_id	bigint		not null	
action	citext_max_6		not null	
action_datetime	timestamp with time zone		not null	
src_pk_column1	citext_max_23		not null	
error_message	citext		not null	

Indexes:  
 "pk\_sync\_error\_doc" PRIMARY KEY, btree (stage\_id, action\_datetime), tablespace "INOW\_Data"

Tablespace: "INOW\_Data"

## SYNC\_HIST\_DOC table

This table contains historical synchronization event details for each execution of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure including performance and throughput metrics and run-time parameters.

Column	Type	Collation	Nullable	Default
sync_datetime	timestamp with time zone		not null	
sync_rpm	integer			
sync_rows	bigint			
sync_usec	numeric			

```

| sync_batch_size | integer | | | | |
| sync_max_mins  | integer | | | | |
| sync_result     | citext_max_16 | | | | |
+-----+-----+-----+-----+-----+
Indexes:
    "pk_sync_hist_doc" PRIMARY KEY, btree (sync_datetime), tablespace "INOW_Data"
Tablespace: "INOW_Data"

```

## SYNC\_STAGE\_DOC table

This table holds information about modified rows in the source table (IN\_DOC).

For every row modified (INSERT, UPDATE, DELETE) in the source table (IN\_DOC) the SYNC\_TRG\_DOC trigger will insert a row into the staging table to stage it for synchronization by the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure.

If document creation volumes are high, this staging table will also grow at the same rate. Be sure to execute the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure periodically to synchronize the new IN\_DOC table (NEW\_DOC) with the current IN\_DOC table.

During the execution of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) procedure, to synchronize data between the source and destination tables, rows are processed in batches according to the batch size and ordered by the stage\_id to promote a first in first out (FIFO) process. After a batch of rows has been successfully processed and committed then the rows will be removed from the staging table.

If all the staged rows have been synchronized, then the staging table will be empty.

```

Table "inuser.sync_stage_doc"
+-----+-----+-----+-----+-----+
| Column | Type | Collation | Nullable | Default |
+-----+-----+-----+-----+-----+
| stage_id | bigint | | not null | |
| src_pk_column1 | citext_max_23 | | not null | |
| src_creation_time | timestamp without time zone | | not null | |
| action | citext_max_6 | | not null | |
| action_datetime | timestamp with time zone | | not null | |
+-----+-----+-----+-----+-----+
Indexes:
    "pk_sync_stage_doc" PRIMARY KEY, btree (stage_id), tablespace "INOW_Data"
Tablespace: "INOW_Data"

```

## SYNC\_STATE\_DOC table

This table contains details for the last synchronization event as performed during the execution of [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) the procedure.

```

Table "inuser.sync_state_doc"
+-----+-----+-----+-----+-----+
| Column | Type | Collation | Nullable | Default |
+-----+-----+-----+-----+-----+
| schema_name | citext_max_128 | | not null | |
| src_table_name | citext_max_128 | | not null | |
| dest_table_name | citext_max_128 | | not null | |
| total_rows_synced | bigint | | not null | 0
| last_sync_datetime | timestamp with time zone | | | |
| last_sync_rows | bigint | | | |
| last_sync_rpm | integer | | | |
| last_sync_batch_size | integer | | | |
| last_sync_usec | numeric | | | |
| last_sync_max_mins | integer | | | |
| last_sync_result | citext_max_16 | | | |
| last_sync_error | citext | | | |
+-----+-----+-----+-----+-----+
Indexes:
    "pk_sync_state_doc" PRIMARY KEY, btree (schema_name, src_table_name), tablespace "INOW_Data"
Tablespace: "INOW_Data"

```

## SYNC\_TRG\_DOC trigger

This database trigger is used to populate the [SYNC\\_STAGE\\_DOC](#) staging table whenever data is inserted, updated, or deleted in the IN\_DOC table. It is used by the synchronization framework to determine which rows have changed in the source table and facilitates the synchronization of rows to the destination table. This DML trigger will fire after each row is inserted or updated or deleted in the source table.

## SYNC\_FUNC\_DOC trigger function

This trigger function is associated with the [SYNC\\_TRG\\_DOC](#) trigger and defines the actions of the trigger when rows are inserted, updated, or deleted into the source table (IN\_DOC).

## SYNC\_SEQ\_DOC sequence

The [SYNC\\_FUNC\\_DOC](#) trigger function uses this sequence to generate the primary key (stage\_id) when populating the [SYNC\\_STAGE\\_DOC](#) table as rows are inserted, updated, or deleted into the source table (IN\_DOC).

Sequence "inuser.sync_seq_doc"						
Type	Start	Minimum	Maximum	Increment	Cycles?	Cache
bigint	1	1	9223372036854775807	1	yes	10000

## Debugging

You can use the `in_debug` parameter for the various procedures if additional debugging is necessary for troubleshooting.

Depending on the procedure, this can result in a very verbose execution that contains state change information and call stack details and other background call details to assist with debugging certain issues.

- To enable or disable debugging you can specify Yes or No for the `in_debug` parameter when executing a procedure.

```
in_debug citext_max_6 := 'YES' -- Enable debugging output during execution
in_debug citext_max_6 := 'NO' -- Disable debugging output during execution
```

## Synchronization benchmarks (EP2 7.5)

The performance of data synchronization between the source and destination tables is affected by many different factors including the availability of various resources or any contention affecting the availability of those resources. This includes processors, memory, disk queueing, extent allocation, row locking, etc.

A significant factor in the performance of data synchronization is the presence of indexes on the destination table. As the uptime steps are completed and all the indexes are created on the destination table, the additional overhead of maintaining those indexes slows down the throughput of the [IN\\_DB\\_UPGRADE\\_SP\\_SYNC\\_TABLE\\_IN\\_DOC](#) synchronization procedure.

**The following synchronization throughput was recorded during testing**

- With only the primary key index on the destination table, which should be in place for efficient row lookup and synchronization.

**Throughput was around 500K Rows Per Minute (RPM)**

- With with the PK index, and NEW\_DOC\_IDX2, and NEW\_DOC\_IDX2\_C indexes only.

**Throughput was around 170K Rows Per Minute (RPM)**

- With all indexes having been created on the destination table.

**Throughput was around 111K Rows Per Minute (RPM)**

Due to the increased overhead and duration of the synchronization process when indexes are in place, it is advised to delay index creation until closer to the scheduled downtime event. You may also conduct your own internal benchmarking to measure the effects and choose to subsequently remove the indexes or the entire synchronization framework until the days or hours leading up to the scheduled downtime.